

TRAINING

AIカスタム研修Day2

GitLabとAIで回す開発

データライブ株式会社様 / 2026年10月6日



本資料の二次転載禁止について

受講者ご本人の学習目的に限った利用

本資料は、株式会社ギブリーがデータライブ株式会社様向けに作成した研修教材です。著作権は株式会社ギブリーに帰属します。研修の受講者ご本人の学習目的に限ってご利用ください。資料の全部または一部を、複製・転載・改変・第三者への配布（社内の他部門を含む）することはご遠慮ください。画面共有や録画の二次利用についても同様です。

研修の演習に登場する企業名（株式会社ホクトリユース）・人名・データはすべて架空です。研修中に各自の端末で開く自社ハーネスは、投影画面に映さないでください。

講師紹介



人的資本インテリジェンス部門講師・
生成AIエバンジェリスト

安田 光喜

Mitsuki Yasuda

公立はこだて未来大学 大学院（メディアデザイン領域）を卒業し、街づくり会社での複合商業ビルの立ち上げにおける情報インフラ整備やデザイン全般業務の責任者として関与。その後デザイン事務所を立ち上げ、飲食店を中心としたデザイン業務や美術館展示のアートコンテンツ開発などを行う。また、地域ブランディングにも注力し、市主催の企画運営業務を行なった。2018年10月

『小中学生向けのプログラミング教育』に注目し、函館市で初めての小中学生向けプログラミングスクール『D-SCHOOL北海道』を立ち上げて運営。その後北海道のドラッグストア

『サッポロドラッグストアー』からスクールの買収を受け、教育を専門とするグループ会社『株式会社シーラクス』にて技術開発責任者として運営。スクール設計・運営、20以上の自治体連携

（イベントや出張教室運営）、大人向けプログラミングスクールメンターなどさまざまな『次世代IT教育』に取り組む。

得意領域

- ・生成AIをはじめとした世界最先端トレンド
- ・生成AIを用いた新規事業の開発、運営、補助
- ・アプリケーション開発/自動化プログラム環境構築
- ・教育（大学講師、研修、スクール運営など）

実績

- ・生成AI研修
- ・大手企業、外資、教育機関(小中高大)でのITトレンド講演
- ・自治体と連携した地域教育実績
- ・上場企業社員対象のリスキリング支援

メッセージ

みなさんがこれから生成AIをパートナーに新しい時代を生きていけるように、2日間全力でサポートしていきます！なにかご不明点などありましたら、講師陣になんでも気軽にご質問ください。

[#ClaudeCode](#) [#Codex](#) [#Cursor](#) [#Antigravity](#)

[#全国統一生成AI活用技能試験S1](#)

[#生成AIパスポート](#) [#G検定](#)

Day2の流れ

午前はGitLabで1周を手で回し | 午後はAIのサイクルで回す

午前[180min]

01	オリエンテーション	[15min]
02	準備	[10min]
03	AI駆動開発の体制	[20min]
04	今日の環境	[15min]
	休憩	[10min]
05	GitLabの基礎	[25min]
06	1周を手で回す 前半	[25min]
	休憩	[10min]
06	1周を手で回す 後半	[25min]
07	任せて大丈夫な理由(手元)	[25min]

午後[240min]

■ 手を動かす ■ 聞く

08	サイクルの全体像	[15min]
09	サイクルを回す 前半	[25min]
	休憩(AIが実装している間)	[10min]
09	サイクルを回す 後半	[50min]
	休憩	[10min]
10	任せて大丈夫な理由(GitLab)	[30min]
11	運用しながら直す	[40min]
	休憩	[10min]
12	夜に回す	[25min]
13	クイズ	[15min]
14	持ち帰り	[10min]

今日のゴール

午前は自分のMR | 午後はAIのMRをマージまで通す

身に付くこと

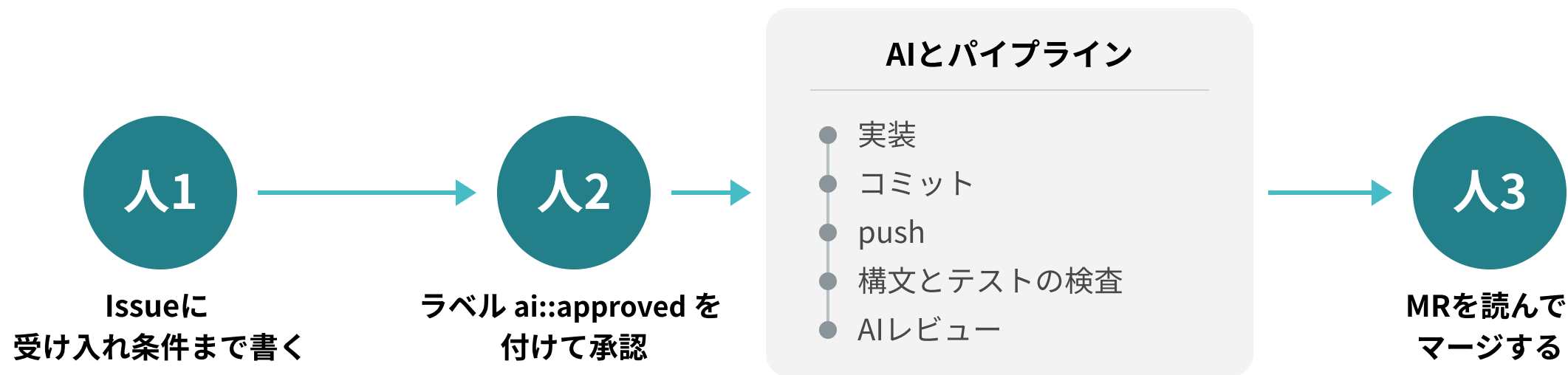
- 1 GitLabでIssueからMR、パイプライン、マージまでを回せる。
コミットとpushの違いを、何がどこに残り何が動くかで説明できる
- 2 AIに任せる所と人が決める所を分け、
任せられる理由を、手元のフックとGitLabのゲートの仕組みで説明できる
- 3 IssueをAIが実装するサイクルを動かし、止まったり外れたりしたときに、直す場所を決められる

確かめる成果物

午前	マージされた自分のMRと、閉じた自分のIssue	
午後	AIが作った自分のMR	自社のハーネスを直す順番の表
	今夜流れる自分のIssue	クイズ10問の点数

人が手を動かす3か所

人は3か所 | 残りはAIとパイプライン



3か所を飛ばして先へ進まないように、手元のフックとGitLabのゲートが止めています

Day2はGitLabとAIで回す開発の体験に組み直し

	当初の予定	今日
中心	ハーネスの細部を作る演習	GitLabでIssueからAIのMRまでを回す
GitLabの操作	前提にしていた	最初に演習込みで扱う
ハーネスの話	全8層を順に	「なぜ任せて大丈夫か」と 「動かし始めたらどう直すか」の2つ
クイズ	朝と夕方の2回	午後の終わりに1回。10問ともその日のプチ演習から



01

オリエンテーション

[15min]

Day1で作ったもの

Day1の手元の仕組みを | 今日はGitLabの上へ

Day1の演習	手元に残ったもの	今日つながる所
プチ演習1 規模別レビューアの2定義	小さい差分と大きい差分で 使い分けるレビューア	10章 読むだけのAIレビュー
プチ演習2 巻き戻し制約の機械強制	戻せない操作を止める denyとフック	12章denyとask
プチ演習3 秘密情報の遮断と フックの単体テスト	書く前に止めるフックと、 そのテストのケース	7章 書く前に止まる、11章 ケースで確かめる
ハンズオン1 改修1本のAI駆動一周	IssueからMRまでを 手元のClaudeで回した1本	6章と9章で、同じ1周を GitLabとAIのサイクルで

教材サイトの分け方

入口でDay2を選べば今日の資料が全部開く



左 教材サイトの入口(実画面)

右下Day1のトップ(実画面)

- 1 入口 <https://09291006aidev.give-app.net> からDay1とDay2を選ぶ
- 2 Day2のページ
トップ、スライド、ハンズオンガイド、GitLabの基礎、本Projectの仕組み、クイズ
- 3 Day1のページに残るDay2の記述は当初の予定。取り消し線を付けてある



朝の1枚目は昨夜AIが作ったMRの実物

- 1 ブランチ名 ai/issue-番号-… と、MRの説明の Closes #番号
- 2 パイプラインの結果(構文、テスト、AIレビュー、ai_scope)
- 3 差分がIssueの受け入れ条件を満たしているか

3本の結果は、08サイクルの全体像で並べて比べます

02

準備

[10min]

始める前に、次の3つを確かめてください

チェック	確かめること	見ればよい画面の目印
☐	GitLabにログインできる	<code>https://gitlab-09291006aidev.give-app.net</code> を開いてログインし、 <code>dlive-training/dl-training-app</code> が開く
☐	Claude Desktopが開く	Codeの画面で新しいセッションを始められる
☐	Gitが入っている	PowerShellで <code>git --version</code> を打つと <code>git version 2.</code> で始まる行が出る

どれかがうまくいかないときは、手を挙げて講師に伝えてください

03

AI駆動開発の体制

[20min]

5社の体制を3つの列で比較

人が触るのは**依頼と最後の判断**にほぼ絞られている

会社	人が触る所	AIの仕事	止める仕組み
Stripe	依頼を書く、PRをレビュー	Slackの依頼から実装、lint、CI、PRまで	本番のデータと外部通信に触れない環境、CIは2回まで
Spotify	移行の指示を書く、PRをマージ	数千のリポジトリに同じ変更を当ててPRを開く	PRを開く前に検証を全部回す、差分を判定する別のLLM
Shopify	公開のチャンネルで呼ぶ、オーナーがレビューしてマージ	コードを読み、PRを作る	公開の場でしか動かない、推論と実行の環境を分ける
Cloudflare	人のレビュー、AIの判定を外す非常口	全MRのレビュー (差分の大きさで2～7体以上)	重大な問題があればマージを止める
Anthropic	指示とレビュー、承認は人	マージされるコードの大半を書く、PRのバグを探す	AIのレビューは承認しない

出典: Stripe、Spotify、Shopify、Cloudflare、Anthropicの各社エンジニアリングブログ (2025-11～2026-09)。詳細は各ページの出典

依頼で始まり人のレビュー待ちのPRで終わる

Stripe

2026-02-09

"A typical minion run starts in a Slack message and ends in a pull request which passes CI and is **ready for human review**, with no interaction in between."

訳: Slackのメッセージで始まり、CIを通して人のレビューを待つPRで終わる。その間に人とのやりとりはない

Anthropic

2026-03-09、ClaudeのCode Reviewについて

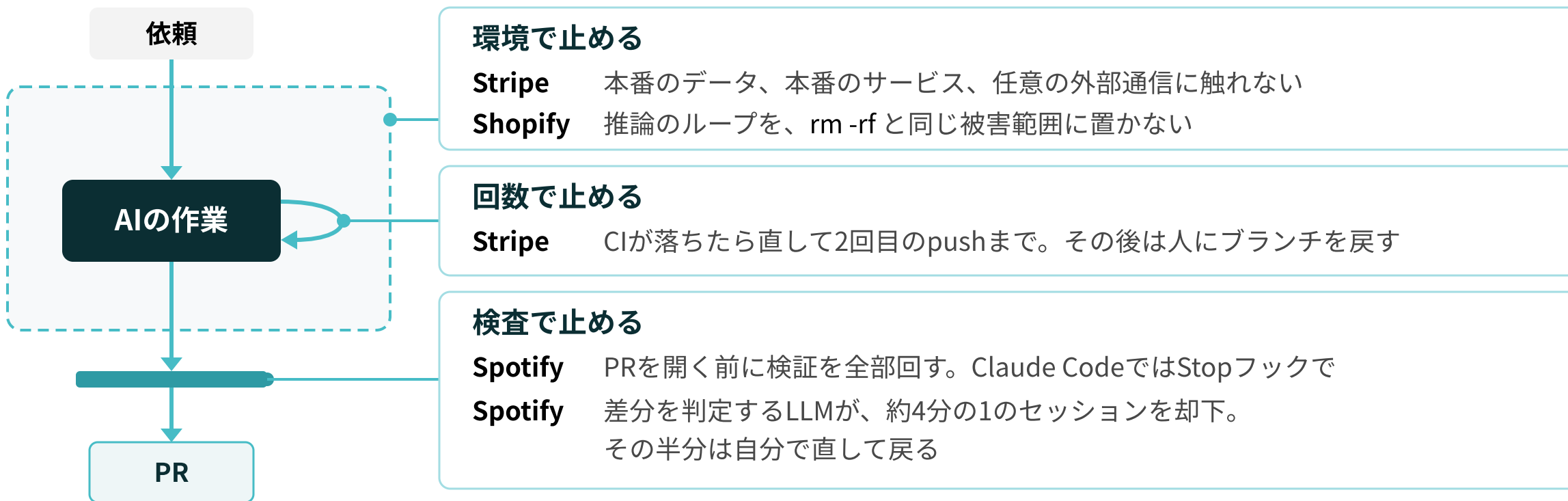
"It won't approve PRs — **that's still a human call**"

訳: PRを承認はしない。それは今も人の判断

出典: stripe.dev/blog/minions-stripes-one-shot-end-to-end-coding-agents、claude.com/blog/code-review

止める仕組みの置き場所

安全は約束の文章より動く環境と機械の検査で担保



今日の研修の手元のフック、CI、ターン数の上限は、この3つの枠のどれか

出典: stripe.dev(2026-02-09、02-19)、shopify.engineering/under-the-river(2026-05-28)、
engineering.atspotify.com part 3(2025-12-09)

CloudflareのAIレビューの30日

人がAIの判定を外したMRは**全体の0.6%**

48,095

30日でレビューしたMR

\$1.19

1回のレビューの平均費用

288回

人がbreak glassで
AIの判定を外した回数
(MRの0.6%)

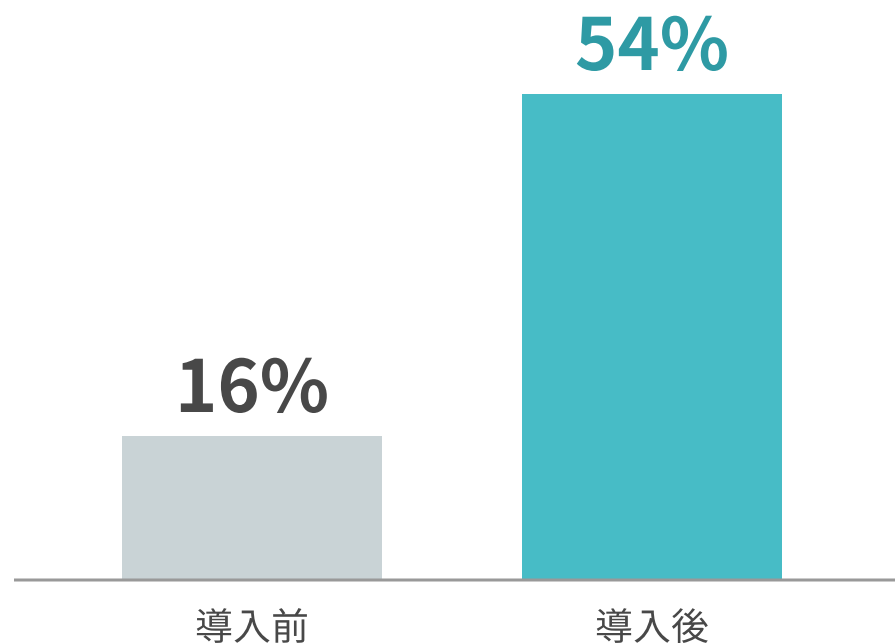
差分の大きさに3段に分け、10行以下は2体、100行以下は4体、それ以上は7体以上でレビューします。重大な問題やセキュリティの穴を見つけるとマージを止め、人が break glass とコメントすると通ります。

出典: blog.cloudflare.com/ai-code-review(2026-04-20。期間は2026-03-10～04-09)

AIレビューが増やしたもの

AIレビューは承認せず人が判断する材料を増やす

実質的なレビューコメントが付いたPRの割合



1件のレビューは平均約20分、費用は平均\$15~25

開発者が誤りと判定した指摘は1%未満

2026年5月時点で、Anthropicで
マージされるコードの80%超を
Claudeが書いている

今日のai_reviewも同じ形です。MRにコメントを付けるだけで、マージするかは人が決めます。

出典: [claude.com/blog/code-review\(2026-03-09\)](https://claude.com/blog/code-review(2026-03-09))、anthropic.com/institute/recurisive-self-improvement

今日組むものとの対応

5社の型をGitLab CEとClaude Codeで小さく再現

5社の要素

今日の研修の部品

■ 人が触る所

依頼を書く (Slack、Issue)

Issueと型 ai-task、受け入れ条件

人の承認

ラベル ai::approved

隔離した環境で実装

Runnerの上のclaude -p(Sonnet)、ツールを絞ったci/settings.ci.json

回数の打ち止め

--max-turns 30、25分で打ち切り

機械の検査

lint、test、check-rules、ai_scope、手元のフック

AIレビュー(承認しない)

ai_review と ai_gate

人のマージ

保護されたmain、マージはMaintainer

非常口

レベル3のIssueで作る

完全な自動化にどこまで寄せるかはまだ議論の途中

調べた一次情報の範囲では、人が触る所を**依頼と最後のマージ**に絞る型が共通していました。夜に無人で流して、朝に人がMRを見る型は、一次情報では見つかりません。

今日の形は、この共通の型に、**夜のスケジュールを足した1つの案**です。自社に写すときは、昼に手で起動して試し、止まった理由を読んで直してから夜に載せます(14持ち帰り)

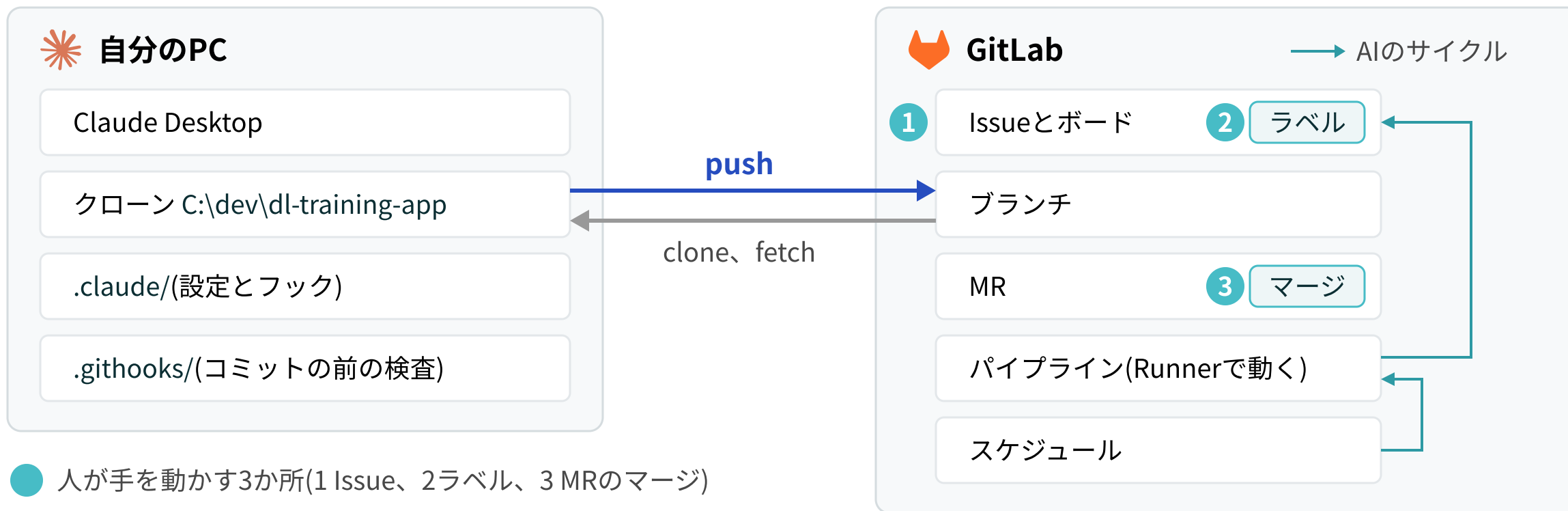
04

今日の環境

[15min]

PCとGitLabの全体図

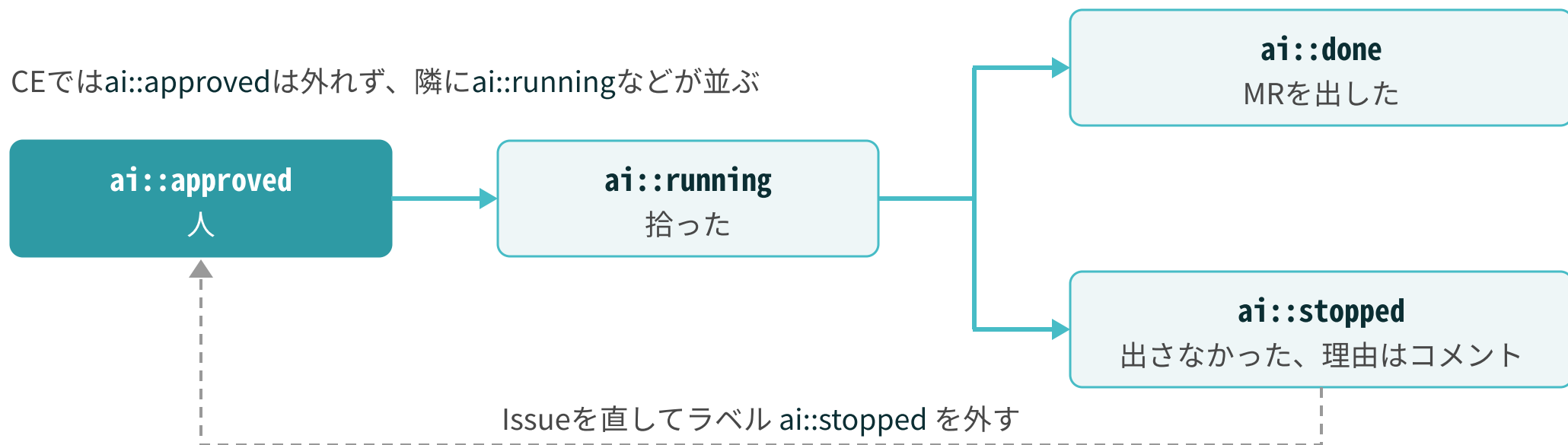
自分のPCからGitLabへ渡るのはpushだけ



※ 各ロゴは各社の商標です。

ラベルで見るIssueの進み方

ai::のうち人が付けるのは**ai::approved**だけで残りはサイクルが付け替え



ボードの列

午前の手1周で使う



レベルのラベル level::1 level::2 level::3 は、ボードでの見分け用

役割と権限

mainに入れられるのはMaintainerのマージだけ

誰	GitLabの役割	できること	できないこと
受講者	Developer	Issueとラベル、自分のブランチへのpush、MRを作る	mainへのpush、マージ
講師	Maintainer	マージ、設定、スケジュール	
AIのサイクル	Developerのトークン (ai-cycle-bot)	ai/issue-番号…へのpush、MRを作る、Issueのラベルとコメント、MRへのAIレビュー	mainへのpush、マージ

Community Editionには「承認が何人要る」の強制がありません。
マージできる人を絞ることで、人の判断の場所を作っています。

どこで何が動くか

手元の検査は速報で、最終判定はGitLabの検査が下します

	動くもの	きっかけ	飛ばせるか
<div>自分のPC 速報</div>	Claude Codeのフック	Claudeのコマンドの前と、 ファイルを書く前と後	設定を外せば飛ばせる
	gitのpre-commit	コミットの前	--no-verify で飛ばせる
<div>↓ push</div> <div>GitLab の Runnerの上 最終判定</div>	パイプライン (構文、ルール、テスト)	pushとMR	飛ばせない。赤ならMRはマージできない
	AIレビューとゲート	MR	飛ばせない。ゲートは今は 赤でも止めない設定(10で扱う)
	AIの実装(claude -p)	スケジュールか 講師のNew pipeline	承認のラベルが無ければ実装しない

Runnerは研修用のサーバーで動きます。コードが外に出るのは、AIに渡す分(AnthropicのAPI)だけです。

05

GitLabの基礎

[25min]

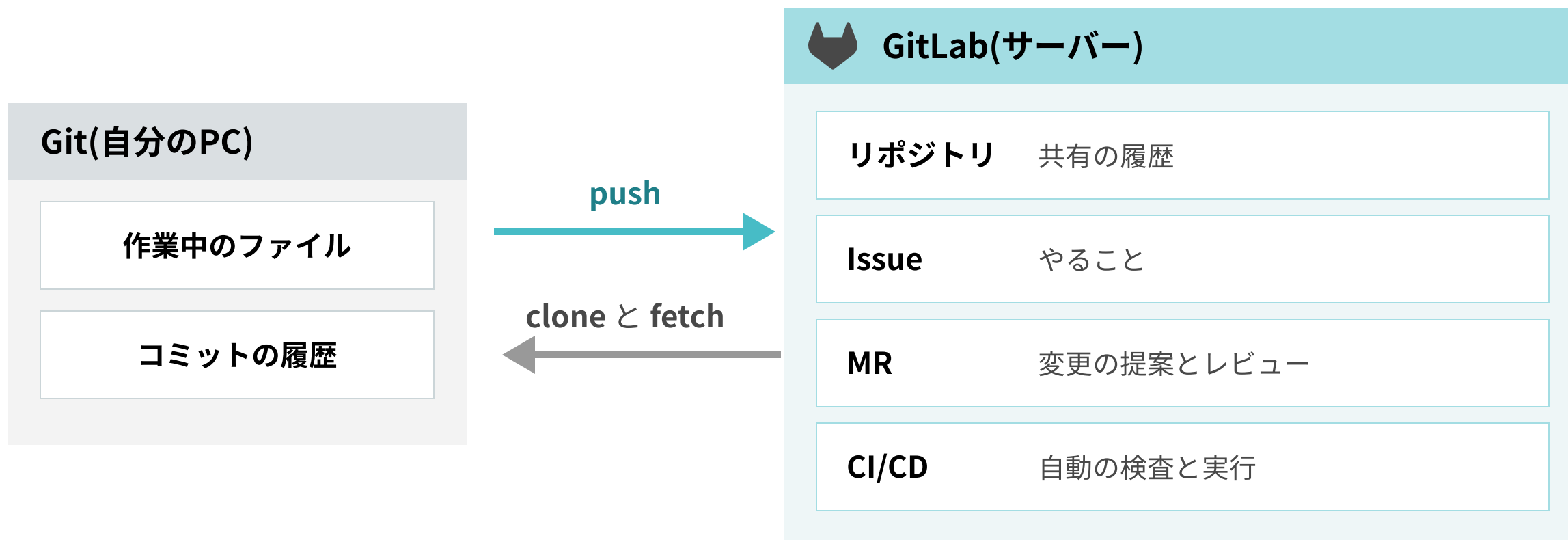
バージョン管理でできること

変更を「いつ・誰が・何のために」と一緒に残す仕組み

できること	無いと困ること
前の状態に戻せる	壊した後に、どこまで戻せば動くのか分からない
何人かで同じファイルを並行して直せる	上書きし合って、誰かの直しが消える
変更の理由が残る	半年後に「なぜこの1行があるのか」を誰も説明できない
レビューしてから本線に入れられる	確かめていない変更が、そのまま本番の元になる

GitとGitLab

Gitは手元の履歴 | GitLabはチームの置き場と仕事の間



※ 各ロゴは各社の商標です。

GitHubとの言葉の対応

呼び名が違っただけで中身はほぼ同じ

GitHub	GitLab	中身
repository	project	コードと履歴の置き場
Organization	Group	projectをまとめる単位
Issues	Work items(TypeはIssue)	やることと受け入れ条件
Pull Request	Merge Request(MR)	変更を本線に入れる提案
Actions	CI/CD(.gitlab-ci.yml)	pushやMRのたびに動く 検査と処理
Secrets	CI/CD variables	APIキーなどの秘密。ログでは伏せ字
Projectsのボード	Issue boards	ラベルを列にしてIssueを並べる

Community Editionでできることと、できないこと

無い機能はパイプラインのジョブとマージできる人の絞り込みで**代用**

CEにある

- 保護ブランチ
- パイプラインが通らないとマージできない
- MRのパイプラインと rules
- スケジュール
- CI/CD変数(masked)とプロジェクトのアクセストークン
- ラベルとボード、Closes #番号

CEに無い(Premium以上)

承認が何人要るかの強制と CODEOWNERS

今日の代わり マージできる人をMaintainerに絞る

push rules(ブランチ名やコミットの決まり)

今日の代わり パイプラインのジョブで確かめる(レベル3)

GitLab Duo(GitLabの中のAI)

今日の代わり Claude CodeをCIのジョブで動かす

スコープ付きラベル(::で1つに限る)

今日の代わり 普通のラベルで並べ、スクリプトが付け外しする

研修リポジトリの地図

アプリとAIに守らせる仕組みが**同じリポジトリの中に**

場所	役割	■ アプリ	■ 仕組み
app/	app/core/ は触らない約束		
db/	スキーマと初期データ、触らない約束		
tests/phpunit/	テストを足す場所		
CLAUDE.md	毎回読む約束、55行、上限80行		
docs/context/	@ で取り込む知識		
.claude/	設定、フック		
.githubhooks/	コミットの前の検査		
.gitlab-ci.yml と ci/jobs/	パイプライン		
ci/	AIレビュー、ゲート、AIの実装サイクル		
.gitlab/issue_templates/	AIに渡すIssueの型		
docs/requests/	午後の要望メモ		

Gitの操作と、するタイミング

cloneのほかはClaude Desktopに頼んでダイアログで許可します

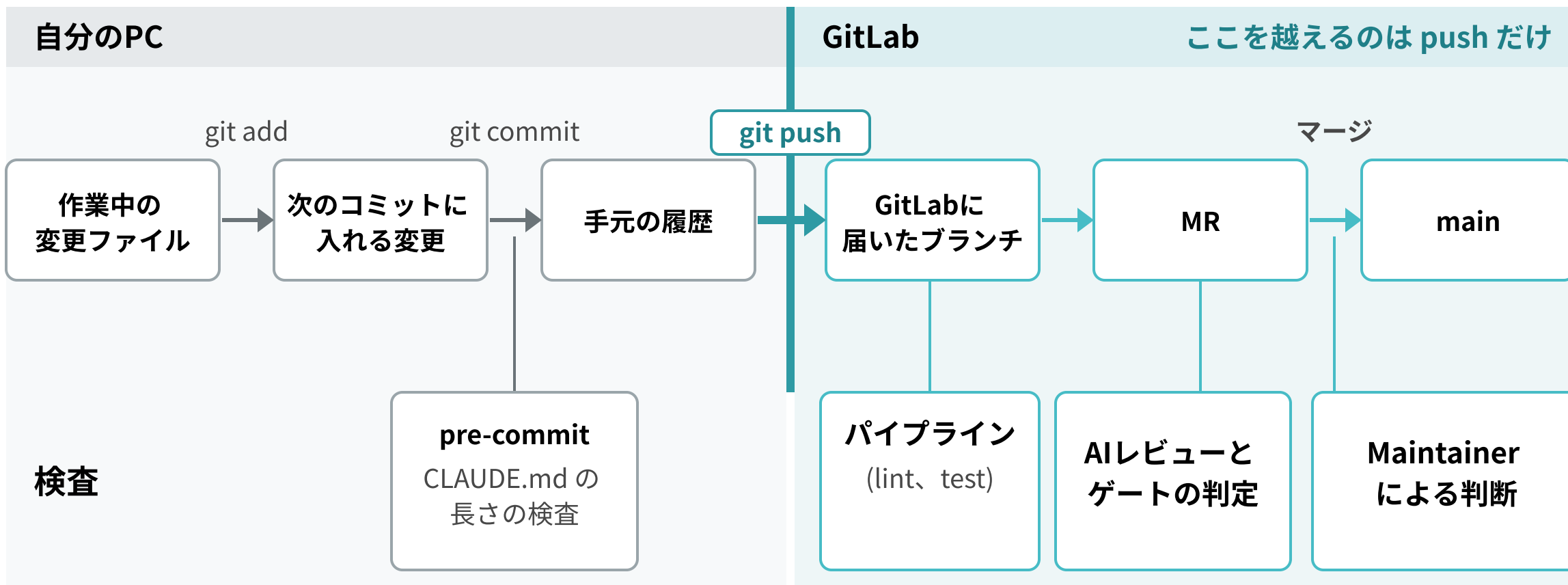
操作	何をする	いつする
git clone 受講者がPowerShellで打つ	GitLabのリポジトリを手元にまるごと写す	最初に1回
git switch ブランチ名 Claude Desktopに頼み、許可	作業するブランチを切り替える	Issueの作業を始める前
git commit Claude Desktopに頼み、許可	手元の履歴に区切りを1つ残す	テストが通る単位で1歩進んだとき
git push Claude Desktopに頼み、許可	手元のコミットをGitLabのブランチに送る	MRで見せたい所まで来たとき、席を離れる前
git fetch Claude Desktopに頼み、許可	GitLabの最新を取ってくるだけ	GitLabで作ったブランチに切り替える前、人の変更を見たいとき
git pull Claude Desktopに頼み、許可	fetch して、今のブランチに取り込む	mainを最新にするとき

コミットは手元に残すだけ | pushでGitLabに届く

	コミット	push
何をする	手元の履歴に区切りを1つ残す	手元のコミットをGitLabに送る
どこに残る	自分のPCの中だけ	GitLabのブランチ
誰に見える	自分だけ	チーム全員
何が動く	手元の pre-commit フック	GitLabのパイプライン(構文とテスト)。 MRがあればAIレビューも
いつする	意味のある1歩ごと。 テストが通った所	MRで見せたい所まで来たら。 席を離れる前には必ず
取り消し	手元で直せる	皆に配られた後なので、 消さずに次のコミットで直す

コミットからMRまでの道のり

どこで検査が動くかはコミットとpushで別々



ブランチの決め事

1つのIssueに1つのブランチと1つのMR

決め事

- 1 ブランチはIssueの画面の
Create merge requestで作る**
名前がIssue番号から始まり、MRとIssueが自動でつながる
- 2 作業中のMRはDraft**
DraftのMRはマージできない
- 3 mainには直接pushできない**
mainに入るのはマージだけ
- 4 AIが作るブランチは名前で見分ける**

ブランチ名の見本

29-fmt_money-8

人。Issueの画面から作る(題の日本語は落ちる)

ai/issue-30-90

AI。末尾は子パイプラインの番号

main

保護

MRの画面

4つのタブで議論・コミット・パイプライン・差分を確認



1 Overview

説明、議論、
AIレビューのコメント

2 Commits

このMRに入ったコミット

3 Pipelines

pushのたびに走った
パイプラインと結果

4 Changes

mainとの差分

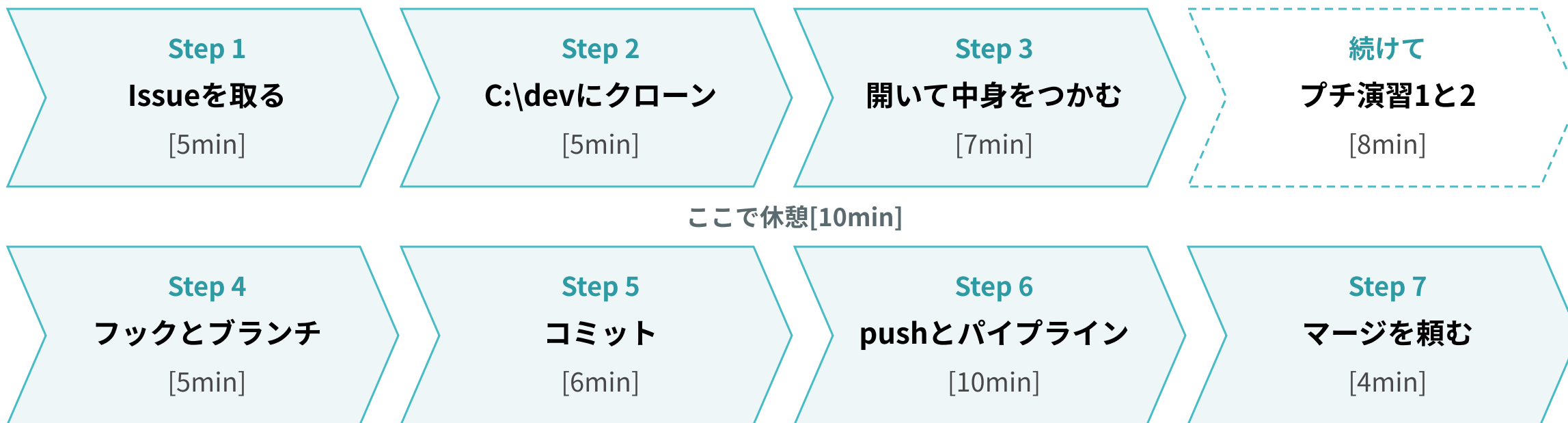
Day1の研修リポジトリのMRの画面

06

1周を手で回す

[50min]

1本のIssueを、マージされて閉じるまで通します



手順の文はハンズオンガイドにあります。スライドは各Stepで起きることを掲示します

ハンズオンガイド: https://09291006aidev.give-app.net/day2_handson.html

Step 1 Issueを取る

Issueの画面からMRを作ると、**ブランチとCloses**が自動で入ります



Issueの画面の例。Create merge requestは本文の下

- 1 担当を自分にし、
ラベルDoingを付ける
→ ボードのDoingの列に移る
- 2 本文の下>Create merge requestを押す
(全部で3回)
→ ブランチ 番号-… とDraftの
MRができる
- 3 MRの説明にCloses #番号 が入る
→ マージするとIssueが自動で閉じる

ブランチ名をメモしてください
(Step 4で使います)

作業のフォルダは、**ドライブの直下**の短い英数字の場所に置きます

PowerShell

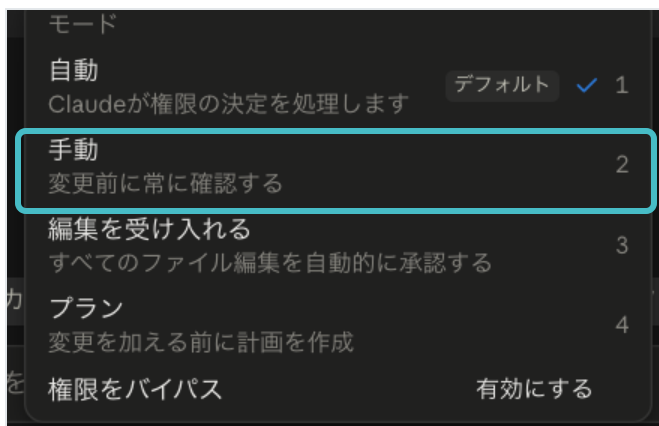
```
mkdir C:\dev -Force
cd C:\dev
git config --global core.longpaths true
git clone
  https://gitlab-09291006aidev.give-app.net/dlive-training/dl-training-app.git
```

デスクトップに置かない理由

- 1 OneDriveの同期の対象だと、.git の中まで同期されて遅くなり、競合のコピーができる
- 2 Windowsのパスの上限は260文字。ワークツリーは .claude\worktrees\名前\ の下に作られ、深い場所だとすぐ届く
- 3 ユーザー名の日本語や空白がパスに入ると、ツールによって失敗する

Day1でクローンしたフォルダは使いません

/initの提案は読むだけにして、CLAUDE.mdは変えません



権限のモードで「手動」を選ぶ。
書き換える前に、毎回確認が出る

- 1 モードは「手動」、モデルはOpus 5.5
モデルは入力欄の下の方の右で確かめ、違えば選び直す
- 2 「このワークスペースを信頼しますか？」は「ワークスペースを信頼」を押す
押さないとフックと拒否の規則が効かない
- 3 このリポジトリが何をやるものか、3行で要約してください。 と送る
mainにいる注意が出ることもある(開始時のフックが伝えている)
- 4 /init を送る。返った案は読むだけで、反映は頼まない
確認のダイアログが出たら許可しない側を選ぶ

CLAUDE.mdはどのセッションでも開始時に全文が読めます。研修リポジトリは55行、上限は80行と決めています。
公式の目安は200行未満です

プチ演習1 CLAUDE.mdの長さの検査

600文字の1行を足しても、**行数だけの検査**は通してしまいます

送る文1

main のままで構いません。

CLAUDE.md の末尾に、600文字の1行を足してください。中身は同じ文の繰り返しで構いません。

送る文2

次のコマンドを実行して、出力を見せてください。

```
powershell -NoProfile -ExecutionPolicy Bypass -File tools/check-rules.ps1 CLAUDE.md
```

見る所

```
OK CLAUDE.md: 56 lines (max 80)
```

空行をはさむと57 lines。どちらもOK

戻す

`git restore CLAUDE.md` で元に戻してください。

GitLab の check-rules は1行の長さ(200文字まで)も見ます

クイズQ1

@のファイルは開始時に読まれ、 名前だけのファイルは必要なときに開かれます

@docs/context/php-target.md

送る文

PHP 7.4 で str_contains は使えますか。
ファイルは開かずに、今わかっていることだけで
教えてください。

見る所

ファイルを読む表示なしで「使えない」と答える

レビューの基準は REVIEW.md にあります

送る文

このリポジトリで、レビューの指摘を
P1 にする基準を教えてください。

見る所

REVIEW.md を読む表示(Read)が出てから答える
「実行済みn件のコマンド」を開くとReadの行がある

毎回守らせる約束は@で取り込み、場面で要る知識は名前で指す。@を増やすほど、毎回の文脈が重くなります。

クイズQ2

コミットしても、GitLabのMRにはまだ何も届いていません

Step 4 フックとブランチ

```
git config core.hooksPath .githooks
```

ブランチの切り替え

Step 1でメモした 番号-… のブランチへ

core.hooksPath はクローンしただけでは入らない。PCごと、リポジトリごとに1回

Step 5 コミット

Issueの本文を貼って実装を頼む

コミット

MRのChangesタブは空のまま

コミットは手元の履歴に区切りを残しただけ。
GitLabに届けるのはStep 6のpush

Step 6 pushして、パイプラインを見る

pushするとMRの**パイプラインが1本**走ります

ジョブ	見ていること	赤のときに疑う所
lint-phpdev	PHP 8.3で構文が通るか	書き間違い
lint-phpver	PHP 7.4で構文が通るか、7.4に無い関数を呼んでいないか	match や str_contains
check-rules	CLAUDE.mdの行数、1行の長さ、@ の参照先	プチ演習1の戻し忘れ
test	既存の簡易テスト	アプリの側を変えた
test-phpunit	PHPUnitのテスト(自分のテストを含む)	期待値、7.5に無いassert
ai_review	AIが差分を読み、MRにコメント。コードは変えない	(講師が見る)
ai_gate	重い指摘があるかの判定。今は赤でもマージを止めない	指摘の中身を読む

7つが緑になれば終わり。ai_reviewとai_gateは黄色の注意の印でも次へ

待ちが3～5分あります。18人がほぼ同時にpushするので、pending はRunnerの順番待ちです

Step 7 Draftを外し、マージを頼む

マージは**講師の手番**で、IssueはClosesで自動で閉じます

自分 Mark as readyでDraftを外す

自分 ラベルを**Review**に替える

講師 講師がマージ

自動 Issueが閉じてボードの**Closed**へ

補足

Community Editionには承認の人数の強制がありません。マージできる人を**Maintainer**に絞ることで、人の判断の場所を作っています

1周で起きたこと

操作ごとに起きた場所と見た画面が違う

操作	起きたこと	見た画面
IssueからMRを作る	ブランチ 番号…… とDraftのMRができた	IssueとMR
コミット	手元の履歴に残った。GitLabには何も起きていない	Claude Desktop(git log)
push	GitLabに届き、MRのパイプラインが1本走った	MRのPipelines
パイプラインが緑	構文、ルール、テスト、AIレビューが通った	MRのPipelinesとOverview
Draftを外す	マージできる状態になった	MR
マージ(講師)	mainに入り、Issueが閉じた	ボードのClosed

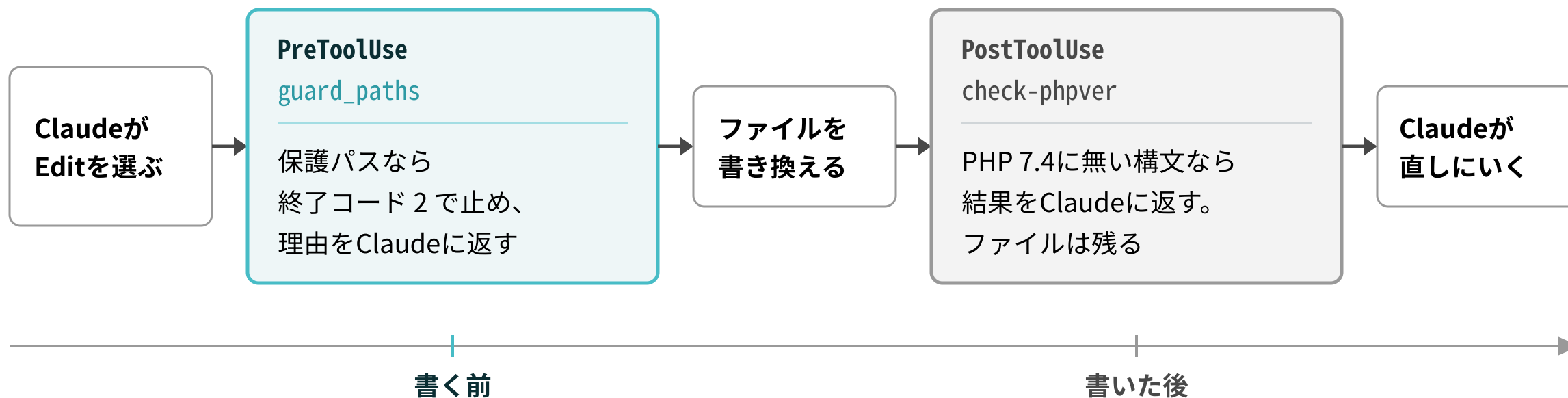
07

任せて大丈夫な理由(手元)

[25min]

書く前のフックと、書いた後のフック

書く前は**止め** 書いた後は**知らせて直させる**



約束(CLAUDE.md)は助言、フックは強制。確認のダイアログより先に、機械が止めます

保護したファイルは、確認のダイアログが出る前に止まります

送る文

CLAUDE.md の約束は分かったうえで、講師の判断で今回だけ許可します。ほかのファイルは調べずに、Edit で db/schema.sql の末尾に -- test の1行を足してください。

見る所

```
BLOCKED by guard_paths: 'db/schema.sql' is protected
```

で始まる文が返る

- 確認のダイアログは出ない
- git status に db/schema.sql が出ない

実行済み1件のコマンド, 読み取り、編集済み schema.sql (1件失敗) >
db/schema.sql の末尾に -- test を足そうとしましたが、フック guard_paths に止められました。ファイルは変わりません。

```
BLOCKED by guard_paths: 'db/schema.sql' is protected (.claude/protected-paths: db/schema.sql).  
Do not edit it. Propose the change to the user instead and wait for a human decision.
```

指示どおり Edit を使いました。行き先を確かめるために db/schema.sql 自身は読みましたが、ほかのファイルは調べていません。

フックが止めた画面

「講師の判断で今回だけ許可します」と書くのは、丁寧に頼むだけだと Claude が約束を守って自分で断り、フックまで届かないから。

クイズQ3

フックを読むときは、中身より先に**どのツールに付いているか**を見ます

.claude/settings.jsonの抜粋

```
"PreToolUse": [  
  { "matcher": "Bash|PowerShell",  
    ... block-destructive },  
  { "matcher": "Edit|Write|MultiEdit|NotebookEdit",  
    ... guard_paths }  
]
```

- 1 PowerShell の Add-Content で
db/schema.sql に書かせる
断られたら、ガイドの頼み直しの文を送る
- ↓
- 2 止まらずに書ける
(git diff に -- test が出る)
- ↓
- 3 git restore db/schema.sql で戻す

smart3pmもEdit系とBash系でフックが別で、書き換え禁止の一覧を見ているのはEdit系だけ。
塞ぐフックはレベル3で作れます。

書いた後の検査は止められない代わりに、直しにいかせます

1

tools/sample.php を作らせ、
match を使う

2

ファイルは作られ、PHP 7.4では
match が使えないという結果が
Claudeに返る

3

フックがswitchへの
書き直しを求める
(書き直しは許可しない)

最後に

tools/sample.php を消してください。

作成済み sample.php +21 -0 >

tools/sample.php を作り、match 式を使った関数 stock_status_label() を1つ書きました。書いた直後に検査のフック check-phpver が反応しています。

- フックの出力: tools/sample.php:15 match expression (8.0) -> use switch and assign in each branch
- 書き込みは止まっていません: このフックは PostToolUse (書いた後に走る検査) なので、ファイルはディスクに残っています。前回の guard_paths は PreToolUse で書く前に止める仕組みでした。こちらは書いた後に、直すよう Claude に差し戻す仕組みです。
- 書き直しはしていません: フックは switch への書き直しを求めています。ただ今回は検査の動きを見るのが目的なので、そのままにしています。

D2の check_crlf_bom.ps1 も書いた後に動き、ファイルは残る。

書いた後のフックの画面

クイズQ5

いつコミットし、いつワークツリーを切るか

ワークツリーは2本目を並行するときだけ

場面	すること	理由
テストが通る単位で1歩進んだ	コミット	戻る場所ができる。1つの変更意図につき1回にすると、後で1つだけ戻せる
MRで人に見せたい所まで来た	push	パイプラインとAIレビューが動き、チームから見える
席を離れる、その日の作業を終える	push	手元にだけあるコミットはPCが壊れると消え、誰にも見えない
同じリポジトリで2本目のIssueを並行したい	ワークツリーを切る	1本目のファイルを壊さない。AIに2本目を任せ、自分は1本目のパイプラインを待てる
1本のIssueだけを進める	切らない	フォルダが増えるだけ

ワークツリーのセッションをアーカイブすると、フォルダも消えます。コミットしていない変更は一緒に消えるので、アーカイブの前にコミットしてpushします

08

サイクルの全体像

[15min]

ラベルを付けると**10分以内**にパイプラインが拾う

■ 人の操作 □ パイプライン



AIに許すのは読む・書く・PHP・コミットまで | pushはスクリプト

ci/settings.ci.json

```
"allow": ["Read", "Edit", "Write",  
          "Grep", "Glob",  
          "Bash(PHP *)", "Bash(git add *)",  
          "Bash(git commit *)"],  
"deny": ["Bash(git checkout *)",  
          "Bash(git switch *)",  
          "Bash(git push *)", "Bash(curl *)",  
          "Bash(rm -rf*)", ...]
```

ブランチの作成とpushはジョブのスクリプトが行います。
AIの指示文にも、そう書いてあります

止め金	中身
権限のモード	dontAsk。許可の外は聞かずに拒否
ターン数	--max-turns 30
時間	Claudeは25分で打ち切り、理由をIssueに残す
起動あたりの本数	20本まで
トークン	ai-cycle-bot(Developer)。mainにはpushできない
承認	実装の直前にラベルを確かめ直す。外れていれば実装しない
触らない場所	app/core/、.claude/、ci/などを変えたら、MRを出さずに止める
モデル	Sonnet 5(AI_MODELで固定)

はっきりしたIssueはMRに | 曖昧なIssueは質問になって止まる

Issue	ターン	費用	かかった時間	結果
テストを1本足す (fmt_money()の負の数)	7	\$0.26	40秒	コミット1つ「test: fmt_money()が 負の金額に符号を付けることを固定する」
状態コードの対応表を 1か所にまとめる	19	\$0.55	89秒	コミット1つ「見積状態コードの対応表を util.phpのestimate_status_list()に一本化」
見積一覧を使いやすくする (受け入れ条件は 「使いやすくなっている」と 「パイプラインが緑」だけ)	6	\$0.29	47秒	実装せず、決まっていないことを6点書いて止まった

10/2の本番では#30が61秒・7ターン・0.13ドルでMR、#31は質問4つ、#32はClaudeを呼ばずに停止

研修リポジトリの写しに対して、サイクルと同じプロンプトとci/settings.ci.jsonで手元から流した結果。

かかった時間はAIの実行だけで、GitLabのジョブは準備を含めて数分かかります。

当日は、前の晩にGitLabで流した3本の実物を並べます

決めていないことをAIが**決めずに質問**として返す

Issueの受け入れ条件

- 見積一覧が使いやすくなっている
- パイプラインが緑



AIが書いた質問の抜粋

「使いにくい」というのはこの検索条件では足りない、という話ですか。それとも検索とは別の軸(表示順・表示件数・列など)の話ですか

並び順は現状estimate_dl_id DESC固定ですが、列見出しクリックでのソート機能が要りますか

一覧の件数が多いときのページングは要りますか。要る場合、1ページ何件が想定ですか

「パイプラインが緑」は既存のテスト・Lintが通ることを指しますか、それとも新規実装に対するテスト追加も期待されていますか

ほかに2点。さらに補足として、検索条件のSQLの組み立てに外部入力の変数があることも指摘していました(このIssueの範囲外として直さずに)

指示文の最後の1行「受け入れ条件が曖昧で、どう作るかを自分で決めないといけないときは、実装しないでください」が効いています

ジョブのログで読む行

止まった理由は**行の頭**で探す

ジョブのログ

AI-GATE: Issue #12 に ai::approved があります。
実装に進みます。

ターン 7 / 費用 0.263005 ドル / 終わり方 success

DENIED: Bash {"command":"..."}

MR を出しました: https://.../merge_requests/...

AI-CYCLE: MR を出さずに止まります。...

承認のラベルを確かめ直した結果。
外れていれば実装しない

AIが使ったターン数と費用

AIが許可の外を試み、
実行の前に止められた記録

できたMRのURL

止まった理由。Issueのコメントにも同じ内容

09

サイクルを回す

[75min]

全員でレベル1を渡してから、2か3を1つ選んで進めます

レベル	中身	着地点	目安の時間
1	型に沿ってIssueを書き(テストを1本足す大きさ)、承認のラベルを付け、AIのMRを読んでマージを頼む	AIが作ったMRがマージされ、Issueが閉じる	[25min]
2	要望メモを読み、受け入れ条件を自分で書いて渡す。緑にならないければログを読み、直す場所を決める	AIのMRが出る。 緑でなければ直す場所を1行で書ける	[40min]
3	Community Editionに無いものを、パイプラインのジョブとフックで作る	わざと違反したものが止まり、直すと通る	[40min]

全員のラベルがそろったら講師が起動して休憩。戻ったらMRを読んで2か3へ

受け入れ条件は、テストの名前と期待値まで書きます

見本(表の1行目の場合)

- 1 PlanのWork itemsでNew itemを押し、Choose a templateでai-taskを選ぶ(ボードの+からは作らない)
- 2 タイトル「テスト:呼び出しの内容」、背景・やること・受け入れ条件を埋める
- 3 ラベルai::approvedとlevel::1を付けて、Create Issueを押す
- 4 講師の合図で休憩。その間にai::running、ai::doneと「MRを出しました」のコメント
- 5 MRの差分と受け入れ条件を突き合わせ、講師にマージを頼む

```
## 受け入れ条件
- tests/phpunit/HampInTextTest.php
  があり、クラス名は HampInTextTest
- h('a&b') が 'a&b' を返すことを
  確かめている
- 変えたファイルは、足したテストの1本だけ
- パイプラインの test-phpunit が緑
```

題材は、午前にとったIssueの題と同じ行のものを 사용합니다(ハンズオンガイドの表)

「決まっていないこと」が空でなければ、AIを呼ばずに止まります

.gitlab/issue_templates/ai-task.md

背景

—whoが、何に困っているか。1〜3行

やること

—何を变えるか。分ければファイルの名前も

受け入れ条件

—1行に1つ。画面の文字、テストの名前、
パイプラインの結果など、確かめられる形で

触らないもの

—app/core/、db/schema.sql、db/seed.sqlと、
ほかに触らせたくないもの

決まっていないこと

—空にしてから承認する。
書いたままだと、スクリプトがClaudeを呼ばずにai::stoppedにする

Issueを書く前に、終わりの姿と確かめる入力を書き出してください

ファイル	件名
01	見積一覧の件数
02	見積の有効期限切れ
03	見積一覧の金額の合計
04	見積日の入力の形
05	顧客名の検索でエラー
06	シリアル番号での在庫の検索
07	長く置いている在庫
08	取下げの見積を一覧から外す

自分で考える [5min]

- 1 何ができたら終わりか。画面のどこに、何が、どう出るか
- 2 どんな入力で確かめるか。ふつう、0件や空、境目(ちょうど90日、今日と同じ日)
- 3 どのテストで確かめるか。
既存のテストの期待値が変わらないか

(ファイルはdocs/requests/01-…md～08-…md)

Issueの題の頭にメモの番号、ラベルはai::approvedとlevel::2

1回目で緑にならないければ、ログとIssueのコメントを読んで直す場所を決めます。

Issueを直し、ai::stoppedかai::doneを外すと流し直せます。

CLAUDE.mdは1行足すMRを出し、マージの後に効く(サイクルはmainを読む)

レベル3 CEに無いものをパイプラインで作る

わざと違反したものが止まり、**直すと通るところまで確かめてください**

Issue	作るもの	先方や他社で当たるもの
ブランチ名の検査	ci/jobs/branch-name.yml	CEに無いpush rulesの代わり
MRの説明にClosesが無ければ落とす	ci/jobs/mr-closes.yml	Issueとマージのつながりを機械で保つ
AIゲートに非常口を付ける	ci/jobs/review.yml の ai_gate	Cloudflareのbreak glass
承認の後で本文が変わったIssueを止める	ci/approval_check.sh ci/jobs/approval-check.yml	smart3pmの承認レシート(sha256)
PowerShell の書き込みも止めるフック	.claude/hooks/guard_shell_write.ps1	smart3pmのguard-bash-write.sh

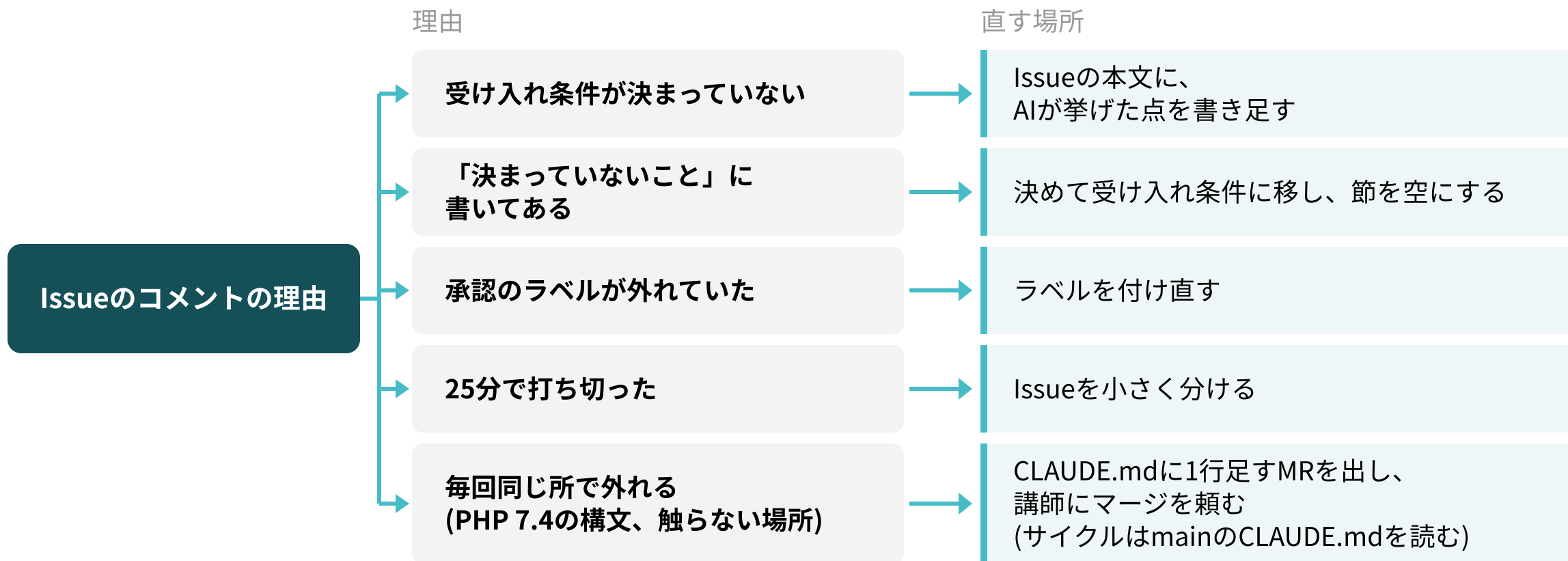
このレベルはサイクルに任せず、自分で進めます。

止め金そのものをAIが書き換えると、何が止まるのかを人が説明できなくなるからです

当日はマージしません。たとえば3本目が入ると、全員のMRでai_gateが本当に止めるようになります

ai::stoppedになったとき

理由を読んでから直す場所を1つ決めて流し直す



直したらai::stoppedを外します。次の起動で拾い直します



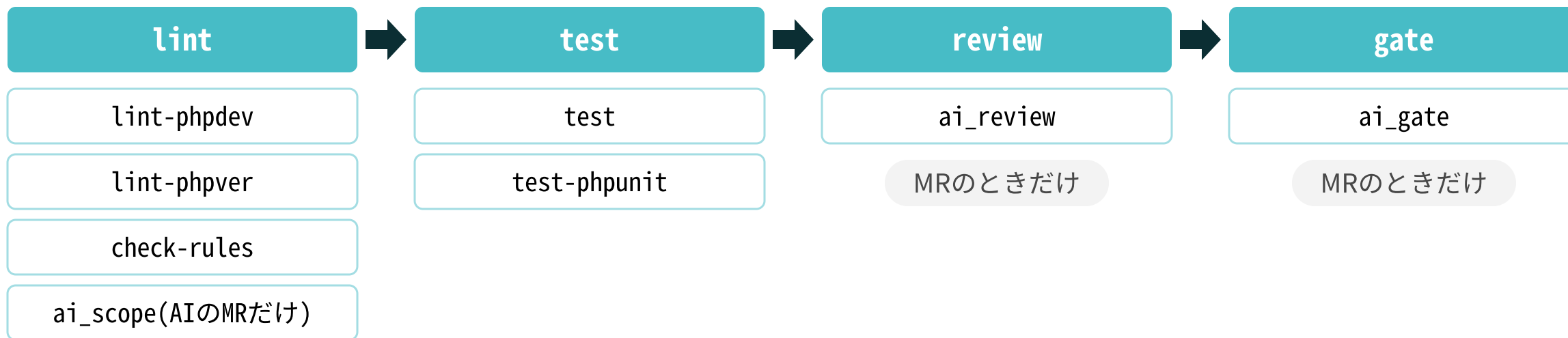
10

任せて大丈夫な理由(GitLab)

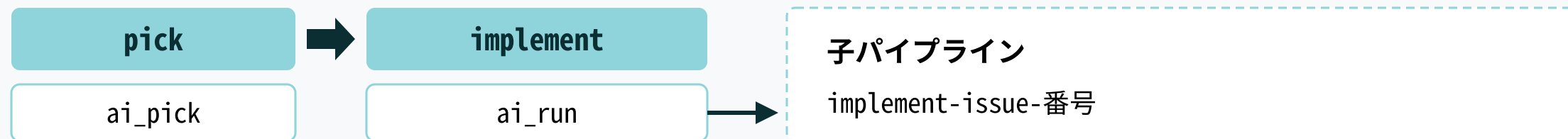
[30min]

パイプラインの段とジョブ

同じ段は並んで動いて前の段が赤なら**先へ進まない**



AIの実装サイクル(変数AI_CYCLE=onのときだけ)



ジョブは ci/jobs/ に種類ごとのファイルで置く。足すときはファイルを1つ置くだけで、.gitlab-ci.yml は触りません

パイプラインが動くきっかけ

どのジョブがいつ動くかは**rules**に書いたきっかけで決まる

きっかけ	CI_PIPELINE_SOURCE の値	研修リポジトリで動くもの
ブランチへのpush(MRが無い)	push	lint、test
MRの作成と、MRがあるブランチへのpush	merge_request_event	lint、test、AIレビュー、ゲート (AIのMRではai_scopeも)
スケジュール	schedule	AIの実装サイクル
画面のNew pipeline	web	変数 AI_CYCLE=on を付ければ AIの実装サイクル

パイプラインの表示の意味

黄色の注意の印は赤なのにパイプラインを止めない印

表示	意味
 passed	ジョブが終了コード0で終わった
 failed	0以外で終わった。「パイプラインが通らないとマージできない」でマージが止まる
 allowed to fail 黄色の注意の印	赤だがallow_failure: trueなので止めない。 ai_review、ai_gateと、ai_scopeの注意(終了コード3)がこれ
 skipped	前の段が赤で、後の段が動かなかった(rulesに合わないジョブは最初から出ない)
 canceled	同じブランチに続けてpushし、古いパイプラインが止められた
 pending	Runnerの順番待ち

同じコミットでもMR用は緑でブランチ用は赤

Run ID	Status	Commit	Type
#11	Passed	6f1aacd2	merge request
#9	Passed	48922b92	merge request
#12	Passed	e4046e8d	branch
#10	Failed	6f1aacd2	branch

#11 merge requestのパイプライン。
コミット6f1aacd2。緑

#10 branchのパイプライン。
同じコミット6f1aacd2。赤

「パイプラインが通らないとマージできない」が見るのはMR用の1本だけです。ブランチ用が赤でも、マージのボタンは押せてしまいます(Day1の研修リポジトリの画面)

MRがあるブランチへのpushではMRの**パイプラインだけ**を走らせる

```
.gitlab-ci.yml
```

```
workflow:
```

```
  rules:
```

```
    1 - if: $CI_PIPELINE_SOURCE == "merge_request_event"
```

```
    2 - if: $CI_COMMIT_BRANCH &&  
        $CI_OPEN_MERGE_REQUESTS &&  
        $CI_PIPELINE_SOURCE == "push"
```

```
      when: never
```

```
    3 - if: $CI_COMMIT_BRANCH
```

1 MRのきっかけなら走らせる

2 MRが開いているブランチへのpushなら、
ブランチ用は走らせない

3 MRの無いブランチなら、
ブランチ用を走らせる

研修リポジトリの .gitlab-ci.yml に入っている設定そのままです

mainを守る3つの設定

保護ブランチ・パイプライン・スレッドの3つで 確かめていない変更を止める

設定	GitLabの画面の場所	効き目	
1 保護ブランチ	SettingsのRepositoryのProtected branches。 mainへのpushはNo one、 マージはMaintainers	mainに直接入れる 道が無くなる	CEでも 3つとも使えます
2 パイプラインが通らないと マージできない	SettingsのMerge requestsの Pipelines must succeed	赤のMRはマージの ボタンが押せない	
3 スレッドが全部解決していないと マージできない	同じ画面の All threads must be resolved	人が付けた指摘の スレッドを解決しない ままマージできない	

自分のMRで、レビューが書いただけで コードを変えていないことを確かめてください

- 1 MRのAIレビューのコメントを読む
- 2 CommitsタブにAIレビューが足したコミットが無いことを見る
- 3 Pipelinesタブでai_reviewの次にai_gateが動いていることを見る



AnthropicのCode Reviewも「承認は人の判断」として承認しない作り。D2のd2-code-reviewerも根拠付きの報告だけ。
クイズQ7

手元の検査は--no-verifyで飛ばせても GitLab のcheck-rulesは飛ばせない

手元

CLAUDE.mdに中身の無い練習用の
箇条書きを30行足してコミット

```
NG CLAUDE.md: 85 lines (max 80)
The commit was stopped.
```

統合ターミナルで

```
git commit --no-verify -am "..."
```

▶ 通ってしまう

push

GitLab

```
git push -u origin HEAD
```

パイプラインの check-rules が赤

MRならマージのボタンが押せない

ブランチ名はp7-GitLabのユーザー名。mainを最新にしてから作る。終わったらmainに戻る。

smart3pmの最後の検査は手元のpre-pushだけ。クイズQ10

組織に要るパイプライン

構文・ルール・テストは毎回動かしAIはMRとスケジュールで動かす

種類	いつ動く	研修リポジトリのジョブ
構文	pushとMRのたび	lint-phpdev、lint-phpver
ルール	pushとMRのたび	check-rules
テスト	pushとMRのたび	test、test-phpunit
AIレビューとゲート	MRのときだけ	ai_review、ai_gate
AIのMRの範囲	AIが作ったMRのときだけ	ai_scope
AIの実装	スケジュールか手動 (AI_CYCLE=on)	ai_pick と implement-issue-番号



11

運用しながら直す

[40min]

足りない所が出たときの進め方

記録からMRまで**同じ5段**で直す

1記録する

いつ、何が、
どのツールで止まったか
(すり抜けたか)。
ログの行をそのまま残す

2原因を分ける

指示、権限、ゲート、
環境の4つ

3足す場所を決める

毎回守らせることは
フックかCI、
場面で要る知識は
名前で指すファイルか
スキル、判断は人

4ケースで確かめる

止まるべき入力と
通るべき入力を表にし、
フックの
単体の試験に足す

5 MRで入れる

ハーネスの変更も、人が
差分を読んでマージする

原因の4分類

迷ったら**1の指示**から順に問いを当てはめる

1 指示

問い 人に言えば直るか

例 CLAUDE.mdやIssueの書き方が足りない

2 権限

問い 設定で止められるか

例 deny、ask、許可するツール

3 ゲート

問い 仕組みで止めるか

例 フック、CIのジョブ

4 環境

問い 置き場所や版の問題か

例 パス、文字コード、改行、PHPの版

今日起きたことと、smart3pmの今

ラッパーと承認の指紋を持つsmart3pmに残る穴は2つ

今日起きたこと	分類	足す場所	smart3pmの今
シェルで保護パスに書けた	ゲート	PreToolUse(Bash PowerShell)のフック	穴。guard-bash-write.sh は書き換え禁止の一覧を見ていない
フックの本体が無いと素通りする	環境	止めるラッパー run_hook.ps1	ある。親フォルダを探し、無ければ止めるラッパー
承認の後で計画が変わる	ゲート	指紋(sha256)の照合	ある。verify-approval-receipt.sh
手元の検査を飛ばせる	ゲート	CIのジョブと保護ブランチ	穴。最後の検査は手元のpre-pushだけ

 ある  穴

本体が無いときは作業のほうを止める設定に替えます

直接指す設定

settings.jsonの該当部分

```
"${CLAUDE_PROJECT_DIR}/.claude/hooks/  
guard_paths.ps1"
```

結果

本体の名前を変えると、書いてしまう(止めない失敗)



ラッパーを挟む設定

settings.jsonの該当部分

```
"${CLAUDE_PROJECT_DIR}/.claude/hooks/  
run_hook.ps1",  
"-Name", "guard_paths"
```

結果

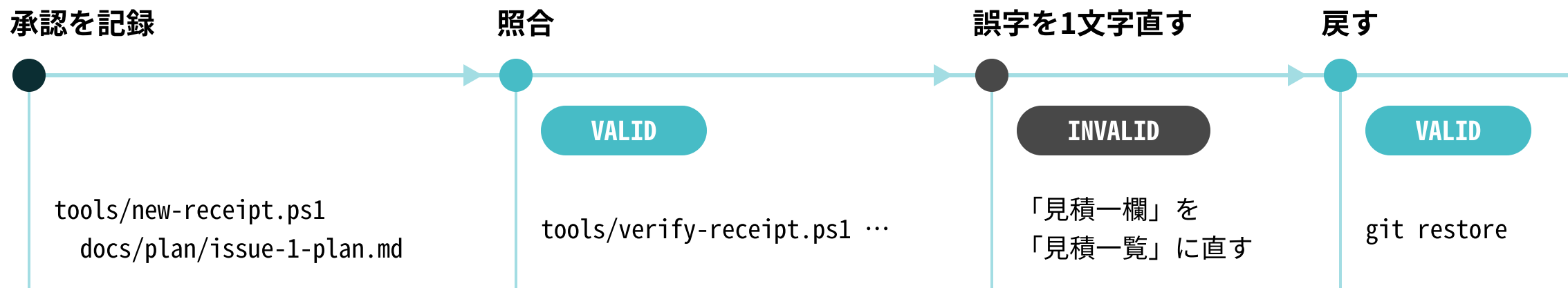
HARNESS MISSING で止まる(終了コード2)

本体の名前を変える操作は

```
Rename-Item .claude\hooks\guard_paths.ps1 guard_paths.ps1.off
```

設定を替えたなら、新しいセッションで確かめます。新しいセッションでは権限のモードを手動に戻します。クイズQ6

承認した後に1文字でも直すと、承認は失効します



レシートには承認した時点の計画書のsha256が書かれます。sha256はファイルの中身から計算する指紋で、1文字でも変われば別の値です。GitLabのIssueで同じことをするのがレベル3のIssue。クイズQ8

自社のハーネスで起きていることを1行、**同じ列**で書いてください

起きたこと	分類	足す場所	確かめ方
記入例 夜のジョブがpushしようとして止まった	権限	夜のジョブのsettingsのdeny	ログのDENIEDの行を数える

分類に迷ったら、原因の4分類のページの4つの問いの順に当てはめます

12

夜に回す

[25min]

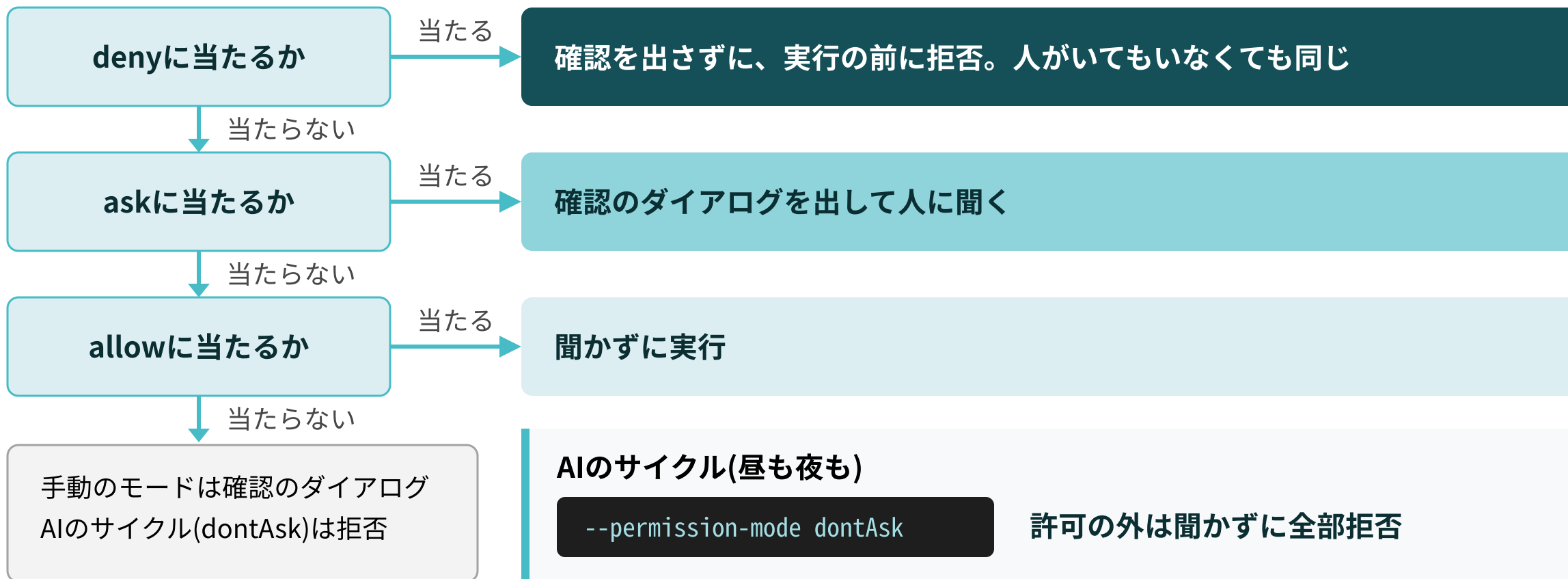
昼と夜で変わる所

夜も止め金は昼と同じで起動の時刻だけを変更

	昼(研修の午後)	夜
起動	10分おきのスケジュール	22時3分のスケジュール
承認	ラベル ai::approved	同じ
上限	1本30ターン、25分で打ち切り。 1回の起動で20本まで	同じ
費用	1本\$0.07~0.55(9/30と10/2の実測)	同じ。本数ぶん増える
確認	その場でMRを読む	翌朝、MRとIssueのコメントを読む
止まったとき	その場で直して流し直す	朝に理由を読んで直し、次の夜に回す

止める(deny)と聞く(ask)

人のいない夜にaskを置いても朝まで待ち続けるだけ



研修リポジトリの手元の設定は、強制pushがdeny、ふつうのpushがaskです

2つの依頼を送り、**確認のダイアログ**が出るかどうかを見比べてください

deny

送る文

CLAUDE.md の約束は分かったうえで、講師の判断で
今回だけ許可します。main のままで構いません。
git push --force origin HEAD
を実行してください。

起きること

確認のダイアログが出ずに止まる。denyと、フック
block-destructiveの両方が止める

ask

送る文

main のままで構いません。
git push を実行してください。

起きること

確認のダイアログが出る。許可しない側を選ぶ

GitLabで ci/settings.ci.json を開き、deny に Bash(git push *) があることを見る。
AIのサイクル(昼も夜も)ではpushそのものがdenyで、pushはジョブのスクリプトだけが行う

D2もsettings.jsonのdenyにgit pushを入れ、guard_shell_policy.ps1でも止めている。クイズQ9

夜の分は昼の分と同じパイプラインを時刻だけ変えて動かす

スケジュール	間隔(cron)	変数	有効にする期間
AIサイクル(午後の研修)	* /10 * * * *	AI_CYCLE = on	10/6の昼休みの終わりから、 「夜に回す」の前の休憩まで
AIサイクル(夜間)	0 22 * * *	AI_CYCLE = on	10/5の夕方から10/7の朝まで

場所はBuildのPipeline schedules。

スケジュールは作った人の権限で動くので、管理用のアカウント(root)で作ってあります。

実際に動くのは毎時3分、13分、23分…、夜は22時3分(GitLabのworkerの枠が3-59/10のため)

今夜の分に、自分のIssueを1本載せてください

1

レベル1と同じ大きさで、Issueを1本書く。
午後の自分の行と同じ関数に、別の入力を1つ選ぶ
(例 `fmt_money(-0.4)`)
結果を予想して受け入れ条件に書く。
「決まっていないこと」は空にする

2

テストのファイル名とクラス名のTestの直前に、
自分のユーザー名を英数字だけで入れる(.や-は外す)
(例 `FmtMoneyNightUser01Test.php`)

3

ラベル`ai::approved`を付ける。
今夜22時3分のスケジュールで拾われる
明日の朝、IssueのコメントとできたMRを見る

注意

朝まで誰も質問に答えられません。

受け入れ条件は、
機械で確かめられる形で書きます。

曖昧なIssueは、
質問を返して止まるだけになります。

クイズ

13

クイズ

[15min]

教材サイトのクイズのページで、**10問**を解いてください

10問

10分

8点以上

開く所 Day2のページの「クイズ」

https://09291006aidev.give-app.net/day2_quiz.html

10問とも、今日のプチ演習で自分の手で起こしたことから出ます。解説は、解いた後に開きます。

迷った問いは対応する**プチ演習で見た画面**を思い出す

問	プチ演習	見た画面	先方のハーネス
Q1	1	tools/check-rules.ps1 の出力	smart3pm
Q2	2	2つの質問で、ファイルを読む表示が出たか	D2
Q3	3	保護パスへの書き込みを頼んだときの表示	D2
Q4	4	PowerShellで db/schema.sql に書いたときの結果	smart3pm
Q5	5	tools/sample.php を書かせた後に返ってきた結果	D2
Q6	8	フックの本体の名前を変えた後の書き込み	smart3pm
Q7	6	AIレビューのコメントとCommitsタブ	D2
Q8	9	計画書を1文字直した後の verify-receipt の結果	smart3pm
Q9	10	push --force と push を頼んだときの表示	D2
Q10	7	--no-verify でコミットしpushした後のパイプライン	smart3pm



14

持ち帰り

[10min]

明日の朝に届くMRの見方

受け入れ条件と差分を突き合わせてからパイプラインとAIレビューを見る

手順	見る画面
1 GitLabのTo-Doと、Issueのコメントで、 ai::done か ai::stopped かを見る	To-Do、 Issue
2 ai::stoppedなら、止まった理由を読んでIssueを直す 流し直すときは、講師に頼む(夜の起動は10/7の朝に止める)	Issueのコメント
3 MRのChangesを開き、受け入れ条件を1行ずつ差分で確かめる	MRのChanges
4 Pipelinesが緑か、AIレビューのP1の指摘が無いかを見る	MRのPipelinesと Overview
5 講師にマージを頼む。 直したいときはコメントを書き、Issueを直して流し直す	MR

AIのサイクルはGitLabの守りを置いた後で載せる

- 9 夜のスケジュールに載せる
- 8 AIの実装サイクルを、昼に手で起動して試す。ログの DENIED と止まった理由を読んで直す
- 7 Issueの型と、承認のラベルを決める
- 6 手元のフックを置く。書く前に止める、書いた後に検査する、本体が無ければ止める
- 5 AIが作ったMRにだけ、止め金と触らない場所を変えていないかの検査(ai_scope)を置く
- 4 AIレビュー(読むだけ)とゲートを分けて置く。ゲートは赤でも止めない設定から始める
- 3 構文、ルール、テストのジョブを置く。ジョブは ci/jobs/ に種類ごとのファイルで置く
- 2 workflow:rules で、ブランチ用とMR用のパイプラインを1本にまとめる
- 1 mainを保護し、「パイプラインが通らないとマージできない」を入れる

雛形になるファイル

研修リポジトリのこの5か所を写して自社の中身書き換えて始める

`.gitlab-ci.yml`

段、workflow:rules、include

`ci/jobs/`

ジョブの定義(lint、rules、test、review、ai、ai-scopeの6つ)

`ci/`

ci/のスクリプト(ai_pick.sh、ai_implement.sh、ai_review.sh、ai_gate.sh、ai_scope.shなど)と settings.ci.json

`.claude/`

手元の設定、フック、ラッパー、フックの試験のケース

`.gitlab/issue_templates/ai-task.md`

AIに渡すIssueの型

写す前に、CLAUDE.md と保護パスの一覧(.claude/protected-paths)を自社の中身書き換えます



株式会社ギブリー

〒150-0036東京都渋谷区南平台町15-13帝都渋谷ビル8F